TABLE NO

STUDENT ID NO

MULTIMEDIA UNIVERSITY®

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2022/2023

TSN3151 – PARALLEL PROCESSING

(All sections / Groups)

12 JULY 2023
9:00 a.m – 11:00 a.m.
(2 Hours)

Q1	
Q2	
Q3	
Q4	
Total	

INSTRUCTIONS TO STUDENT

1. This is an Open Book Exam. You are allowed to bring in any edition of *Parallel Programming* by Barry Wilkinson & Michael Allen, and/or 10 sides of A4 self-typed notes stapled together.
2. This Question paper consists of 10 pages with 4 Questions only.
2. Attempt **ALL** questions. The distribution of the marks for each question is given.
3. Please write all your answers in this question paper clearly.

Question 1 [25 marks]

a) Using the e-cube routing (also known as a deadlock free routing algorithm), determine the route taken in a 6-dimensional Hypercube network from Node 45 to node 56. Your final answer should be stated in Node (decimal) number; show your steps clearly. Use the same method we practiced in the tutorials. Show all your steps clearly for full marks. [11 marks]

Continued...

b) Lintrix Bhd. possesses a supercomputer that comprises 65,536 processors and is currently facing a task that requires significant number-crunching. The algorithm involved in this task does not necessitate the neighbouring computations to communicate with each other. However, the time required for data transmission over the network is observed to be 45% of the time required for computation on average. This is mainly because each computation is swift but involves handling a vast amount of data.

Considering these circumstances, would the best approach to implement this task be MPI, or OpenMP? It is important to justify the choice with an explanation, taking into account why the other approach is not as suitable for this task. [7 marks]

Which method:
Explanation:

c) BFG Chong Enterprises Sdn. Bhd. has a network of 32 PCs, and each PC has 16 processors. They need to perform a significant number-crunching job, which doesn't require the computations of neighbouring processors to communicate with each other. The average time for data transmission over the network is 0.05% of the time needed for computation.

Given these circumstances, which parallel processing approach would be most suitable: MPI or OpenMP? Please provide a justification for your choice and explain why the other approaches is not as appropriate. [7 marks]

Which method:
Explanation:

Continued...

Question 2 [25 marks]

Listing 1 is an MPI program:

```
1  #include <stdio.h>
2  #include "mpi.h"
3
4  int main(int argc, char**argv){
5    int p, np, a, b, c, d;
6    MPI_Init(&argc, &argv);
7    MPI_Comm_rank(MPI_COMM_WORLD, &p);
8    MPI_Comm_size(MPI_COMM_WORLD, &np);
9
10   switch (p){
11     case 0: a=1; b=2; c=3; d=4;
12       MPI_Bcast(&a, 1, MPI_INT, 3, MPI_COMM_WORLD );
13       MPI_Allreduce(&b,&d,1,MPI_INT,MPI_SUM,MPI_COMM_WORLD);
14     break;
15
16     case 1: a=5; b=6; c=7; d=8;
17       MPI_Bcast(&b, 1, MPI_INT, 3, MPI_COMM_WORLD );
18       MPI_Allreduce(&b,&d,1,MPI_INT,MPI_SUM,MPI_COMM_WORLD);
19     break;
20
21     case 2: a=9; b=10; c=11; d=12;
22       MPI_Bcast(&d, 1, MPI_INT, 3, MPI_COMM_WORLD );
23       MPI_Allreduce(&b,&d,1,MPI_INT,MPI_SUM,MPI_COMM_WORLD);
24     break;
25
26     case 3: a=13; b=14; c=15; d=16;
27       MPI_Bcast(&a, 1, MPI_INT, 3, MPI_COMM_WORLD );
28       MPI_Allreduce(&b,&d,1,MPI_INT,MPI_SUM,MPI_COMM_WORLD);
29     break;
30   }
31
32   printf("Process %d: a:%d b:%d c:%d d:%d\n",p,a,b,c,d);
33   MPI_Finalize();
34   return 0;
35 }
```

Listing 1

a) What is the output of *Listing 1*? [6 marks]

```
Process 0: a: ____ b: ____ c: ____ d: ____
Process 1: a: ____ b: ____ c: ____ d: ____
Process 2: a: ____ b: ____ c: ____ d: ____
Process 3: a: ____ b: ____ c: ____ d: ____
```

Continued...

b) Rewrite *Listing 1* using only MPI_Send and MPI_Recv to do exactly the same thing. You do not need to recopy the lines that don't change – just refer to the line numbers, e.g. "Lines 1-8" and then just write line 9. [19 marks]

Continued...

Note: Don't worry if you don't need so much space; I'm allowing for just in case people have big handwriting.

Continued...

Question 3 [25 marks]

Listing 2 is a serial program:

```
 1  #include <stdio.h>
 2  #define SIZE 82139
 3  #define START 3.25
 4  #define STEP 0.75
 5  #define END ((SIZE*STEP)+START)
 6
 7  double f1(double x);
 8  double f2(double x);
 9  double f3(double x);
10  double f4(double x);
11
12  int main(int argc, char *argv[])
13  { double total=0, x;
14    FILE *infile = fopen("datafile.txt","r");
15    int i;
16
17    for (x=START; x<END; x+=STEP)
18    { fscanf(infile,"%d",&i);
19      switch(i)
20          { case 1: total += f1(x); break;
21            case 2: total += f2(x); break;
22            case 3: total += f3(x); break;
23            case 4: total += f4(x);
24          }
25    }
26    fclose(infile);
27    printf("The result of the calculation is %f\n", total);
28  }
```

Listing 2

Assuming that f1, f2, f3, and f4 are defined, convert *Listing 2* into an OpenMP program that will speed up the calculations by utilising all the available processors.

Continued...

You do not need to recopy the lines that don't change – just refer to the line numbers, e.g. "Lines 1-15" and then just write line 16. [25 marks]

Continued...

Question 4 [25 marks]

Listing 3 is a serial program:

```
1  #include <stdio.h>
2  #define SIZE 82139
3
4  double f(double x);
5
6  int main(int argc, char *argv[])
7  { double total=0, x;
8    FILE *infile = fopen("datafile.txt","r");
9    int i;
10
11   for (i=0; i<SIZE; i++)
12   { fscanf(infile,"%lf",&x);
13     total += f(x);
14   }
15   fclose(infile);
16   printf("The result of the calculation is %f\n", total);
17 }
```

Listing 3

Convert *Listing 3* into an MPI program that would distribute the work in parallel properly. Assume that the function f is provided. [25 marks]

Continued...

End of Paper